MYTHS AND REALITIES:

Defining Re-engineering for a Large Organization

By Sandra Yin and Julia McCreary
Internal Revenue Service
8405 Colesville Rd., Suite 300
Silver Spring, MD 20910-3312

MYTHS AND REALITIES:
Defining Re-engineering for a Large Organization

Abstract

- I. Introduction
  - A. Internal Revenue Service Tax Systems Modernization
  - B. Myths and Realities
- II. Concepts, Content and Context: IRS Assessment and R^3 Methods
  - A. Objectives
  - B. Software Re-engineering Taxonomy
  - C A Framework for Redevelopment
  - D. Portfolio Analysis
  - E Up, Over, and Down: Applied Use of Methods
  - F. Technical Opportunities
  - G. Project Plans for Four Executive-Selected IRS Systems
- III. Tips, Tricks & Traps: The R^3 Prototype
  - A. Objectives
  - B. Findings
  - C Technical Approach for Data
  - D. Technical Approach for Process
- IV. Features and Futures: Automated Tool Market Survey
  - A. Objectives
  - B. Interview Results: IRS R^3 Objectives
  - C Tool Integration
  - D. Multi-Vendor R^3 Tool Set Examples
  - E Commercially Available Integration Environments
  - F. Transition Challenge: Managing Multiple Tools
  - G. Roll Up Your Shirt Sleeves: R^3
- V. Business Re-engineering
  - A. How does Technology Support Business Needs?
  - B. How does Software Re-engineering Support Business Re-engineering?
  - C Risk Management
- VI. Recommendations for the Organization
  - A. Define and Implement an Infrastructure
  - B. Prepare an Inventory of Current Systems
  - C Conduct a Business Needs Assessment/Measurement Phase
  - D. Develop Criteria to Select R^3 Candidates
  - E Write a Transition Strategy
    - 1. Two sets of transition strategies
    - 2. Prioritize R^3 objectives for the entire enterprise
    - 3. Use R^3 opportunities to support organizational readiness
  - F. Choose a Candidate Project(s) Targeting Implementation
  - G. Establish an R^3 Team
  - H. Procure Tools which meet the Business Needs, Objectives and Timeframes
  - I. Rationalize Enterprise-wide Data and Standardize Data Names
- VII. Technology Transition
- VIII. Marketing R^3 Internally
- IX. Realities of R^3

Introduction

This paper describes the background and results of three studies concerning software reverse engineering, re-engineering and reuse (R^3) hosted by the Internal Revenue Service[1] in 1991 and 1992. The situation at the Internal Revenue - aging, piecemeal computer systems and outdated technology maintained by a large staff - is familiar to many institutions, especially among management information systems. IRS is distinctive for the sheer magnitude and diversity of its problems: the country's tax records are processed using assembly language and COBOL, spread across tape and network DBMS files, all crying out for a better way of doing business! How do we proceed with replacing legacy systems? The three software re-engineering studies looked at methods, CASE tool support, and performed a prototype project using re-engineering methods and tools. During the course of these projects, we discovered critical issues broader than the mechanical definitions of methods and tool technology. If we could all just develop new software from scratch, life would be so simple. Therein lies the thorn for most large scale "new" development: planning an orderly transition, organizational readiness and business re-engineering.

Tax Systems Modernization

The IRS is in the process of modernizing all of its tax processing systems. Tax Systems Modernization (TSM) is a long-term effort to move from stand-alone legacy business systems built on old technology to integrated systems based on enterprise-wide planning and management. Lines of code for IRS existing systems has been quoted to number 16 million - 8M Unisys COBOL and 8M IBM assembler. Though often cited, the statistic is not revealing without further classifications and other measurements or breakdowns. These programs are used for mainframe processing of taxpayer accounts and tax returns in regional IRS centers. IRS also has other systems written in IBM COBOL, C, and 4GL. All parties agree - manually maintaining these systems with often overlapping functionality comes at great cost. The planned modernization effort will take the better part of the decade and, since 1989, has been primarily approached by the Service as a new, top-down development effort using information engineering.

In 1991 and 1992, the IRS undertook three projects to answer the questions: "Is re-engineering a technology which could assist the IRS in its modernization effort?" and, if so, "How should we proceed?". The scope of all three projects was software reverse engineering, re-engineering and reuse, which we refer to as R^3 internally at IRS and within this paper. The projects were initiated by the Integration Division of Information Systems Development of the IRS National Office. Partners included the Transition Management Office, which is presently defining the Software Development Environment (SDE) for the modernization, and the Compliance Division, which maintains systems controlling tax law compliance cases, both within Information Systems Management. The projects were contracted out to three private companies. Within IRS, these projects were defined and managed by a small group of people who were active in the work and R^3 questions as posed to the vendors.

[1] The Internal Revenue Service (IRS) is a federal agency belonging to the United States Department of Treasury. Its mission is to administer and enforce the federal tax law for The United States of America.

Myths and Realities

With the experience of the methods, tools and prototype projects behind us, combined with lessons learned in information engineering and watching CASE technology improve over three years, we can see that some of the initial questions and expectations for R^3 reflected a naivete about what CASE can do to support software engineering. We shall describe some "myths and realities" we discovered in the context of related subjects. Here are some mythical, yet widely-held, organizational assumptions surrounding the R^3 projects at the onset.[2]

- Reverse engineering and re-engineering are synonymous.
- Re-engineering soils the pure top-down engineering effort.
- The old programs are so encrusted with history there is nothing to salvage.
- Re-engineering is fully automated.
- A single CASE tool is the solution for new development.
- Buy a CASE tool; and don't bother with infrastructure, work process, organizational readiness and process improvement needs.
- Wishful thinking makes it so.

Concepts, Content and Context: IRS Assessment and R^3 Methods

Objectives

The first project undertaken, contracted to Price Waterhouse from the Fall of 1991 through mid-1992, was intended as a means of assessing IRS re-engineering opportunities and setting clear objectives for implementation. Re-engineering terms were defined in a taxonomy and a methodology for implementation was proposed based on IRS needs. The initial objectives were as follows:

- Produce a taxonomy of industry-standard R3 terms and definitions.
- Develop and publish taxonomy-supported reverse engineering and re-engineering methods which map to IRS standard methodologies: the existing Software Development Life Cycle (circa '82 Yourdon) IRS standards; and the then-draft Information Engineering Life Cycle (IELC, James Martin's IE Methodology), the IRS standard for modernization development.
- Review IRS documents stating modernization goals, standards, etc. Perform corporate assessment on broad needs and areas of opportunity.
- Connect user-stated needs and objectives with planned processes and tools.
- Further assess and produce sample plans for four executive-selected, IRS systems which represent a wide spectrum of R^3 needs and potential.

Software Re-engineering Taxonomy (Standard Definitions)

It is very important that every organization establish its standard definitions. Standard terms and definitions serve to articulate objectives and tasks, and remove ambiguity from legal documents such as contract specifications. We recommend reusing meanings known industry-wide for a given word, and refrain from creating a specialized definition for your company.

[2] The authors do not believe the any of the myths listed in the paper. We use "mythical" here to mean: without foundation in fact; imaginary; fictitious. Early on during our projects defining re-engineering, simple closer inspection and experience dispelled most myths.

One of the stumbling blocks we encountered from the beginning was the proliferation of "RE-" words, causing confusion rather than providing a basis for clear communication.[3] The IRS documented multiple sources of definitions for a volume of eighty-four terms, and standardized on a single, preferred meaning for each term. IRS generally uses the IEEE definitions, which are often equivalent to industry standards. Here are some examples of definitions for R^3.

Reverse engineering:

The process of deriving a conceptual description of a system's components from its physical-level description, with the aid of automated tools (McClure).

Re-Engineering:

1. A combination of tools and techniques that facilitate the analysis, improvement, redesign and reuse of existing software systems to support changing information requirements (Ulrich).
2. Combined processes encompassing reverse and forward engineering, resulting in a "new" system (IEEE).
3. Improving current systems without impacting current functions, technical platforms or archictectures (Guide).

Reuse:

1. Applying knowledge about one system to another system.
2. Sharing software components, requirements, and effort of maintenance.

While IRS began by endorsing the IEEE definition of re-engineering, we have moved toward a definition to include: using tools to make analyses and discoveries about an existing system and using the tools to create an improved system. The reason is that re-engineering entails so much more than reverse plus forward engineering. Much of the benefit of re-engineering is derived from the assessment, conditioning and positioning activities, discussed in the framework below. We have spent a lot of time and effort in damage control just explaining the definition of re-engineering, trying to move beyond the myths. Meanwhile, today we find across the industry the term **redevelopment** has emerged as a recent de facto standard term describing what IRS calls re-engineering.

A Framework for Redevelopment (Methodology)

The taxonomy was applied to a structure of categories or sequences, to guide work processes or methods. There are numerous frameworks we have seen which provide umbrellas over the taxonomy, all the "re-" words and more. Trade magazines and conferences can provide much information for comparison about what techniques are available, but full-blown details of methods are often proprietary and only available as services rendered during the course of a contracted redevelopment project. Some vendors offer redevelopment services bundled with their proprietary tool.

[3] There exists a plethora of "re-" words and terms: restructuring, redesign, resystemization, re-architecting, re-documentation, redevelopment, re-engineering, reinventing-the-wheel. It is personally frustrating to find every common technical term rediscovered and prefixed with "re-". For several of the "re-" words, we venture the epistemology comes from those with a top-down view of software engineering. Those who think software engineering is evolutionary might think change or improvement is expected, and not require prefixes to describe necessarily iterative tasks. Get beyond "re-" words by using descriptive phases to state objectives, i.e. DBMS migration from IDMS to DB2 vs. re-architecting.

One example of applied methods is the Framework for Redevelopment (Ulrich), reference Slide 9. This framework is the basis for a commercially-available off-the-shelf encyclopedia of objective-driven and "scenario-based" methods, approaches and procedures. It uses Zachman's Information Systems Architecture Framework: enables decision making during Inventory/Analysis; reconciles top-down planning with bottom-up mapping during Functional Assessment; makes source level improvements during Positioning; targets existing maintenance environment or Transformation. The Transformation methods are determined by the implementation scenarios. Transformation options include targeting the Information Engineering's Design and Construction phases in an I-CASE environment. This framework example is noteworthy. It is a new COTS product since Summer, 1992. Its timing indicates the market is demanding scenario-based options to development, and in particular, it offers several missing pieces from top-down information engineering concerning the role of legacy systems in new development and the transition from legacy systems to the new target.[4]

Terminology will vary among methodologies. The methods need to be supported by automated tools; and the terminology and framework - along your organization's objective-driven, scenario-based path - describe the functionality of those tools. A general awareness of the tools on the market needs to be taken into consideration before deciding your objectives. Obviously, there is a trade-off to keep in mind so that you select objectives which may be supported using CASE and which are implemented within time and budget.

Some aspects of the life cycle are only feasible with tool support. Today, methods which work at the source or design level are feasible using tools (e.g. tool-supported measurements or improvements may be made to existing systems which target maintenance). However, we are continually challenged with the question, "What about business functions?" After all, what is the objective in an archaelogical dig into legacy systems if one can not extract the business function? Presently, the tools on the market which work with process logic do not "extract" essential business rules to the analysis level; the analyst works interactively at the design level. Business functions represent requirements. If one has traceability to requirements for an existing system, the redevelopment and transition to a new target would be more straight forward. Necessity is the Mother of Invention. To the vendor community: we need redevelopment support at the analysis level; i.e. ability to extract functional flow (Phemister) and business rules (Gane), and better data/process/object rationalization and management support.

Presently, redevelopment methods and tools are evolving rapidly. Generational advances in hardware architectures and software engineering are also affecting redevelopment methods. We expect near-term evolution of CASE to include: state transition, extensions to entity-relationship modeling, object-oriented methods and

[4] Objective-driven and scenario-based simply refer to letting the parameters of your specific, current environment along with custom needs and goals dictate the solution. This sounds obvious, like common sense. However, some organizations in an early stage of adopting new technology might attempt purist approaches. Originally at IRS, re-engineering was heresy against information engineering methodology, while recently people are more open to possibilities of a symbiotic relationship.

targets, links to client-server tools, etc. Redevelopment tools will need to incorporate these new CASE targets.

Redevelopment work is a team effort, sometimes compared to an archaelogical dig. The technology itself is no "silver bullet". Regardless of how you acquire the methodology suitable for your organization, create teams which include people with prior experience in your redevelopment scenario and in the application domain.

Portfolio Analysis

Common to every framework is a strong, up-front emphasis on "portfolio analysis", assessments to guide decisions about project objectives and strategies. Decisions regarding the benefit of re-engineering any system must always be made in relation to organizational goals and objectives. Portfolio analysis assesses the condition of an existing system or systems, in order to strategically select re-engineering options which are both technically feasible and advantageous from a business perspective, reference Slide 10.

This is management by triage. At an enterprise level, each program maps to the portfolio analysis quadrants, offering a snapshot of the state the enterprise. The enterprise is then grouped into categories of programs with like characteristics, e.g. mission-critical, multiple applications with common functions, hardware platforms. For example, portfolio analysis measures the level of effort to move systems en masse, from mainframe Unisys batch/flat file COBOL to CASE/ client- server target, while pointing out those programs which are not worth the effort.

At a program level, the data structures and processes need to be assessed regarding how well they support business functions and the degree of platform independence. Their quality depends upon how stable or reusable the software components are and whether they represent requirements for the replacement system. Depending upon the combinations of low/high technical value and low/high functional value: retire or rewrite some systems; migrate some intact functionally to new architectures; repair others to last until replacement; and reuse as basis for new development or use as mapping for Current Systems Analysis.

During the IRS project, the corporate assessment was performed "manually", by interviewing people in information systems maintenance and new development, and by reviewing various documents. As a manual effort, the portfolio analysis needs to be "maintained" and become more comprehensive in the future. If an organization performs this current systems assessment with the aid of tools, there is the added benefit of having an authoritative source and integrated inventory (a repository of current operations, systems, platforms, locations, versions, languages, DBMSs, teleprocessing, methods, metrics, measurements, etc.) which could be put to other uses.

Myth: "Metrics and measurements are just unnecessary overhead."
Reality: One can not begin to understand the scope of a problem unless it is measured. Fixes based on anything less is like practicing folk medicine. Show potential customers the use and benefits of each measurement.

Again and again, we cannot stress enough: make informed decisions regarding business objectives before jumping in headfirst. Assessment based on organizational goals determines your selection of projects; project objectives and scenarios determine your methods; and methods determine the appropriate technology you need to get the job done.

Up, Over, and Down: Applied Use of Methods

It is a great temptation to use methods to defend the use of a favorite tool or technology. We experienced a mid-life-cycle crisis in the prototype project when we discovered that the goal of taking the old code into a CASE tool and moving forward was inappropriate for the candidate we had chosen. This is not to say that applying these methods to a project cannot be a good one. Rather, the portfolio analysis, categorizing the system and establishing project goals, was not fully performed.

Be sure to use methods sensibly: still working within a framework and having selected the objectives, the software process and methods should be repeatable for a given scenario-type. The scenario example described above, populating an I-CASE tool with existing system components and using the I-CASE tool to forward engineer to the target environment, can be very beneficial when applied to an appropriate candidate, reference Slide 11.

Note, in the slide, the black arrows between construction and design phases, grey arrows between design and analysis, and white between planning and analysis on the forward engineering side only. This means several things. First, lack of tool support on the process side only makes reverse engineering feasible for creating design-level abstractions from source code. Second, better tool support for the data side allows the possibility to extract analysis phase constructs, through a multi-step process, from the source data structures and data definition language. Third, integration between planning and analysis is manual, top-down only. One's choice of methods, and how "high" up the I-CASE tool to target for reverse engineering, must be dependent on the availability of tool support, the risk taken by that approach, and the level of effort, time and cost. Consider as part of the cost, not only the price of the CASE tools, but the likelihood of needing help from experienced consultants.

Myth: "Reverse engineering means using an automated tool which does everything unassisted upon pressing a button," or, the opposite, "There is no such thing as reverse engineering".

Reality: Let the buyer beware of smoke and mirrors. There are good tools in most all classifications[5]; however, code reverse engineering to an analysis level is not here today. Reverse engineering and re-engineering is hard work, where the analyst is usually assisted in decision-making with interactive tools. Tools in the hands of an inexperienced or undisciplined user can potentially cause damage faster than that user could with bare hands. Make plans, and understand the methods. "Just-in-time training" and mixed teams of specialists and learners can ease technology transfer in the organization.

[5] Tool classes parallel the methods. The following tool classes were created during the Market Survey project and are: existing systems (enhancement, assessment and conditioning); repository load/enhancement; new/replacement systems; repositories; integrated tool set environments; testing/validation; software/project management.

Another methodology provided by Price Waterhouse, reference Slide 12.A, demonstrates re-engineering phases within a life cycle. Here re-engineering tasks can be related to the familiar forward engineering life cycle phases of analysis, design and construction. Examples of applying re-engineering solutions to business problems, reference Slides 12.B-D, are technical redesign, functional enhancement, system rationalization, hardware platform conversion, and CASE migration.

Technical Opportunities
Reuse provides the greatest potential benefits from reverse and re-engineering. Whether in the context of recycling old system components for one-time use by new development, sharing software components between both existing and new systems, or consolidating redundant software components (requirements, data, its behavior and testing, etc.) into a shared resource, reuse has demonstrated benefits. The IRS Assessment and R^3 Methods study included private industry case studies where reuse was universally stated by companies as a principle benefit. During the IRS assessment of re-engineering objectives and targets, the consensus of many interviewees was reuse provided the incentive for re-engineering. The IRS has identified certain functions which could benefit from sharing software components, as opposed to the present way of independently programming and testing some redundant requirements and functions.

Myth: "Reuse is a matter of organizing a reuse library..."
Reality: Reuse is much more than the catalog of reusable components. Reusable components need to be both used and maintained. Give clear thought to the objectives and strategy for reuse. At the highest (software life cycle) level possible, try to put boundaries around the generalized domain in question, and re-engineer or forward engineer from there. Reusable requirements are perhaps the highest objective. Multiple users of a shared software component might add their extensions. Determine procedures for reuse management. Who owns, uses and maintains software components? How to make reusable objects "living and accessible" (are they being used)? Know a given reusable object's "bandwidth" of reusability through a life cycle. What is the reuse repository or SDE framework service information model; etc.? Reuse is a program requiring support and incentives throughout the culture of the organization, from managers to the users of reuse. Reuse the lessons of case studies rather than reinventing the wheel.

Myth: "I am not going to re-engineer, I am using I-CASE [new, top-down development]."
Reality: Re-engineering is not an alternative to I-CASE, competing for devlopment attention. Rather, it supports top-down development at several strategic points, to include verification & validation, current systems assessment and transition issues. An organization with business or management information includes data among its most valuable assets. Any replacement system for MIS must consider the transition and continuity of the data. At a minimum, the developer must include data re-engineering in any development, and better to do so with foresight and a methodology than when it comes time for migrating production data. The developer must examine a system's external data

interfaces.[6] Regardless of how archaic the legacy system may be, the developer of the replacement system needs to know the mapping from old files to new, while the implied data model may be used as a skeletal starting point for new redesign and development. There are tools which make improvements to software components, though often the traceability history between old and new is lost or kept manually. The reality is with or without tool support, the developer must manage the transition.

Project Plans for Four Executive-Selected IRS Systems

Plans were drawn up for re-engineering four systems which were selected because they represented typical and significant software and/or business problems at IRS. Their approaches were seen as repeatable for other similar projects. The four projects were:

1. Share software components - reuse penalty and interest calculations.
2. Convert from Unisys network DBMS to IBM Relational DBMS.
3. Migrate a recently-developed system (electronic returns filing), with a long expected life, to an I-CASE tool; and maintain from the design level of the tool. Reuse some tax form field validation routines between electronic filing and paper returns processing systems.
4. Use tools to work with IBM assembly code (IRS Master File). Options include improving the assembly language code, establishing reusable modules (D-Sects), modeling the Master File data structures, testing conversion to COBOL or I-CASE.

Tips,Tricks & Traps: The R^3 Prototype

Objectives

At the same time the Assessment and Methods project was going on, a prototype was developed by Integrated Microcomputer Services, Inc. (IMS) in Rockville, Maryland. The system was fairly representative of IRS code. It produces letters and correspondences mailed to taxpayers. It had 70,000 lines of COBOL code; 37 batch and online transaction programs; flat files and DMS-1100 network DBMS; Unisys 1100 platform. A contractor lead a mixed team of consultants and IRS staff. The IRS staff had no prior experience with CASE tools, but were knowledgeable as maintenance programmers of that application system. The objectives of the prototype were as follows:

- Reverse engineer existing system into a CASE tool
  - Condition code to be acceptable to CASE tool
  - Decouple data from logic
  - Create design structures from code
  - Create design structures from data
- Normalize/rationalize data model
  - Reconcile multiple entries for logically equivalent objects to a single, standard data name
  - Re-engineer data to conceptual entity-relationship diagram, analysis model

[6] For example, if a system has multiple programs including both flat files and DBMS, we recommend first working with data structures for the DBMS and for the I/O going to/from another system.

- Forward engineer normalized design
- Make modifications to code in the CASE tool
- Forward engineer the whole system to construction, returning to the original platform

We felt this prototype had many of the characteristics of "typical" IRS target environments and would demonstrate benefits and/or difficulties in using CASE technology for maintenance. This prototype had appeared to be a portrait of a re-engineering exercise. The code was conditioned prior to being put into the CASE tool; data was separated from process in the code; data was rationalized and normalized. The challenges we subsequently faced were not the result of the conditioning process. Rather we had neglected to perform an important planning step - setting objectives on the basis of inventory/analysis and needs assessment. The business need should be established first, then the technical mechanics second.

Findings
As a technical proof-of-concept, the prototype was largely successful. It was good experience and valuable lessons were learned. Some of the lessons learned could have been learned by listening to someone else's story like you are doing here today. Let me lay some of those out here.

The candidate system was selected because of its representative size and technical characteristics. It also was fairly well contained, not requiring intricate interfaces with other programs or systems. It had an enthusiastic sponsor: upper and middle managers who authorized the technicians to participate in the 28-week project. The internal characteristics of the system were not so suitable to re-engineering. Unfortunately, the architecture of the system - with its 37 batch and online transaction programs, accessing both flat and DBMS files - was not an appropriate candidate given the pre-selected re-engineering objectives for the proof-of-concept. The candidate system and target objectives were selected without the benefit of Inventory/Analysis and Positioning.

It was assumed the resultant system would look much like the original system did and testing requirements were established based on those assumptions. Half-way through the project, the normalized data model was presented for review and the light went on. The original system had only auxiliary files being accessed through the data base.[7] The substance of the business data was stored on flat files, primarily on tape. The normalized data model, if implemented, would so dramatically affect the processes that the code would have to be abandoned and rewritten. At this point, it became clear that the project objectives had not been carefully selected. Had the objective of the project been to modify the system to its most efficient form (see business re-engineering below), there would have been no question as to how to proceed. We would have followed through with the normalized data model and rewritten all of the code, boiled down to a handful of new programs. However, rewriting the code would demonstrate nothing about reverse engineering the logic into a CASE tool and forward engineering (generating code for) the same functionality to the target

[7] The DBMS was storing the transactions to process later in batch mode against the flat file which held the meaningful data.

platform. At that time, demonstrating the equivalent functionality of business re-engineering in the testing phase would have been difficult; we had already prepared to test 37 programs. We did not have the staff or the time to create equal testing scenarios for the newly developed code, to exactly match the original code.

IRS decided at this mid-way point, rather than forward engineering the fully normalized data model, to instead implement small modifications to the data structures, so that the code could be preserved in close to its original form. This system was neither business re-engineering nor the same as production. In the process of re-engineering the code we discovered some awkward programming styles and processing indicating patchwork, which were preserved into the CASE tool. We realized this candidate system, although tested, should not be put in production.

We can draw valuable conclusions from the work that was done. The project was completed to run on the Unisys, with successful code generation, tracing to design level in a CASE tool. We were able to populate the CASE tool with reuseable objects. Just the learning process of cleaning up the data and building models alone was a valuable one for the technicians.

But more important than these simple technical objectives were the findings which frame the plans for the next re-engineering project. For example, candidate systems need to be chosen based on current systems inventory/assessments which are made in light of business objectives. Also, prototypes should target production. It changes how customers and project sponsors feel about project results. Targeting production also helps focus project objectives and maintain project momentum. Another key component of any re-engineering project is cleaning up the data before populating a CASE tool. This step may seem self-evident to the casual observer, but it is one for which we did not budget time. Important lesson learned. Another lesson entailed facing the reality that a good deal of the work was manual or tool-assisted. Some unique situations may require consulting services or specialized tools which serve a market niche, most of which are only available with contracting services. For example, because the original and target platform was Unisys, there were additional steps at both the front and back end of the re-engineering process, which can be automated with a special purpose tool (i.e. not available for mass market). The technical approach we used on this prototype, mixing manual and tool-asssisted processes, is well-documented and could be reused again with a like project.

Technical Approach: Data Re-Engineering
Slide 16 shows the technical approach we took during the project, listing tools, automated processes and manual processes. A text editor was used to standardize names and Bachman/Analyst™ was used to normalize the data model. Here lies the mid-life-cycle crisis. If we forward engineered the normalized conceptual data model, the flat files disappear, and so would 30 batch programs. Although, getting rid of unnecessary programs is a good idea, this was not the project's originally stated objectives. So, the data model was de-normalized to target most of the original file structures before moving it to the Knowledgeware ADW™ tool where it was merged with the process model. You will see two branches into the ADW tool. The direct link to the ADW Analysis Workbench tool was the entity-relationship data model, as normalized in Bachman. There was a second branch into the ADW Design

Workbench tool. That branch shows the IDMS DDL passing the de-normalized design to the Design Workbench to be merged with the process model.

Technical Approach: Application Re-Engineering
Slide 17 is a diagram showing the process of the application re-engineering. This procedure was fairly straight forward. The imsCASE™ tool[8] was used to perform the special modifications that had to be made to the code in the conditioning phase: removing Unisys distinctives within the code, and modifying the code to reflect data name standardization and new data structures. The imsCASE tool exported system-wide repository objects to a single encyclopedia in the Knowledgeware ADW Design Workbench. As originally specified, analysts made enhancements to the function of the system in the ADW tool to demonstrate maintenance proof-of-concept. Then the ADW Construction Workbench generated code and database manipulation language (DML). On the back end, the imsCASE tool was used to modify code before returning it to the Unisys platform.

Features and Futures: Automated Tool Market Survey

Objectives
The third study, a Market Survey conducted by Case Associates, Inc., developed evaluation criteria for tools and identified tools which meet IRS corporate needs. The Market Survey's objectives were:

- Use re-engineering methodology as a baseline for developing tool classifications.
- Interview representatives from the IRS maintenance and development community concerning objectives behind tool needs.
- Develop and apply tool evaluation criteria across an existing vendor database of several hundred tools.
- Select tool classes of special interest, and determine state-of-art best-in-class. Integrated tool sets were identified as a special tool class.
- Interview top 15 vendors (of the best-in-class) to discuss tools in more details and discuss tool futures.
- Write a white paper on software engineering standards and tool integration.

Interview results: IRS R^3 Objectives
The vendor for Market Survey conducted more interviews.[9] The results appear like a demographic breakdown of an organization in transition. The diverse IRS community falls into camps of five main objectives. These objectives represent five important ways that the IRS can apply re-engineering technology. Everyone can

[8] The ImsCASE tool belongs to the consultant for this contract. It was used for the following purposes on this project: parse/export Unisys COBOL Procedure and Data Divisions; parse/export Unisys DMS1100 Data Definition Language; create repository of system-wide objects including shared subroutines and standard data elements; static analysis of source code and source cross-referencing; and repository data interfaces with Knowledgeware's ADW.

[9] Six months earlier similar re-engineering briefings and interviews had been given by Price Waterhouse for the R^3 Assessment and Methods task.

benefit from the number one, but each of the others are defended as primary objective by one of the organizational units within the IRS. In truth, they provide the parameters for setting target objectives and outlining a transition strategy. Those five primary objectives are as follows:

1. Creation of an inventory of all current systems.
2. Faster software maintenance while capturing information about the current systems.
3. Current system verification and validation of Business Area Analyses created by the Information Engineering group.
4. Information extracted from current systems to assist in the development of:
   - New Business Area Analysis (BAA deliverables in IE); and
   - R^3 I-CASE repository for forward engineering.
5. Transition to new target environments supporting the Standards-Based Architecture (open systems).

Tool Integration

Myth: "CASE tools are presently non-integratable; we should stick to one tool, for the sake of standards and control."

Reality: Unfortunately, our many transition needs and activities are not so simple. One tool does not encompass all the functionality of a software development environment. The goals of the IRS modernization are the driving factors. We look to vendors with non-proprietary interfaces and open systems for the potential to integrate CASE tools.

The goal of tool integration is to make a collection of tools appear to users as a single tool. This aspect of the re-engineering industry is only beginning to produce interface requirements to the vendor community. As stated elsewhere, tool integration is a cry which we, as users, have raised through our acquisition documents. Tools should have the same look and feel (user interface) so that the transition between tools is smooth and learning time is minimal. All the data produced by one tool should be usable by another tool without loss of meaning (semantics) or loss of content. It should be easy to navigate between the integrated tools, carrying the information between the tools. Tool execution should be controllable in a uniform way so tools can be combined to form higher-order functional units.

In a multi-tool, multi-vendor, multi-platform environment, standardized tool integration support is needed to ensure a consistent user interface, common database and standard communication mechanism between tools. An IPSE, Integrated Project Support Environment, is a rich solution, yet unavailable today. It will support such users as tool builders, project managers and application developers to manage software development in an open architecture environment. IRS is watching the CASE industry to see the extent of vendor cooperation on integration standards, and to assess the maturity or likelihood of integrated Software Engineering Environment (SEE)[10] or IPSE support for the future.

[10] The Integrated SEE framework is based on the NIST/ECMA Reference Model (National Institute of Standards and Technology / European Common Manufacturers Association -149) which in turn references the PCTE standard for a Portable Common Tool Environment, a.k.a. "the toaster model".

There are four categories by which one integrates tools: process, presentation, data and control. The SEE is a framework for software engineering, an open systems environment presentation, control and data integration of software engineering tools using standard interfaces and architecture. These standards are still only goals for most vendors, and only partially implemented in all cases. Data integration is regularly offered by vendors. But data integration comes in various "sizes" and is most effective when coupled with control integration. The most complete form of integration, data sharing, is presently only available within single-vendor Integrated CASE tools. Data interchange is offered between a small number of firms, while data linkage, passing flat ASCII files through import/export facilities, is offered by many. The important control mechanisms which give the data meaning are excluded in the transfer in all but the data sharing example. The industry has a long way to go in providing this level of integration.

To define clearly what we, as the user community, are asking of the vendors: process and presentation integration is mandatory - data and control services are optional behind them. The framework for integrating R^3, the open systems environment of IPSE and IEEE is the most important component for integration.

Transition Challenge: Managing Multiple Tools
Reference Slide 24, this is a Venn Diagram representing the information stored by tools supporting current systems (A), cross-systems integration and corporate-level inventories (C), and ICASE development (E), and the intersections among them.[11] This diagram portrays the integration challenge. Using a manufacturing analogy, the three large triangles illustrate a warehouse scenario. The A/B/D triangle makes up a manufacturing supplier. The B/C/D/F triangle represents the storage warehouse. The D/E/F triangle is the manufacturing facility. Where as the first facility produces parts to be used in the final product, their goal is to ship the product to the manufacturer. As often happens, there is an interim shipping transfer point, where raw materials are stored, perhaps even inventoried for use later. The final destination, however, is the manufacurer. In the CASE tools scenario, the A/B/D triangle are the niche tools, and existing systems maintenance, working with primarily construction objects, producing normalized data models or conditioned code. The B/C/D/F triangle represents the integrator, presently held by repository-like tools. The D/E/F triangle is the forward engineering CASE tool, generating code for new or replacement systems. The intersections between the triangles provide the most interesting challenges. There exist interfaces and integration between tools, most of which are proprietary "hot links" between individual "business partners". The integration goals are:

1. Get tools for current systems to allow work on incremental improvements.
2. Take deliverables from A and integrate via C with E.
3. Add value-added management information in C, to synchronize and control enterprise operations.

[11] A = Current Systems, all existing production
B = Change requests, testing, field operations, database admin
C = The Integrator - Configuration Mgt, Enterprise & Management Info
D = Requirements, Rationalized Data/Code, CSA, data migration
E = ICASE development, forward engineering
F = ICASE verification & validation, model management

Multi-Vendor R3 Tool Set Examples

In the review of some of the tool set recommendations, some months later, it is clear that the industry is rapidly changing. At least one of the tools listed here, reference Slide 23, was marketed by a company that is no longer in business. The market survey results were given to us with a caveat: the industry is changing so quickly, recommendations are valid for three to six months. Within twelve months, some change is evident in fully 75% of the tools listed in the survey. Either feature offerings have shifted their position among competitors or companies have diversified. These industry realities confirm the earlier recommendation, only procure tools for a defined need to use in a current project. Don't buy a "tool box" full of tools for use someday.

These tool sets also demonstrate the "hot links" between specific tools. Although this is often marketed as integration, and can be extremely useful if a project requires two tools to communicate, be aware of the limitations of what is being sold. These linkages will need to be maintained, if the project has any extended life. Which vendor will maintain this linkage? What happens if one or the other company goes out of business? We are still waiting for the interchange standards to be implemented.

The optimum goal would no doubt be to wait five years until the industry settled down a bit. But we have immediate problems requiring immediate answers. For this reason, we recommend that technology refreshment clauses be included in new procurements. Identify specific business objectives which can be served by categories of re-engineering tools and acquire those tools for use in clearly defined projects. Or contract with an agency that will provide the products for the work. Consider settling for "good enough", slightly less functionality in the short term using a canned, open product which buys you a vertically upgradable and integratable environment over the long run. Watch industry alliances, national and international standards, and determine which tools position the organiztion with the best options.

Commercially Available Integration Environments

Integration environments are beginning to be available in the marketplace, reference Slide 22. Most of are based on the "toaster model", offering plug-in features for multiple tools. One should look carefully to determine the level of integration being demonstrated. About all "data integration" means today is exchanging entity-relationship diagram or data dictionary information. Know the process by which the tool provides ERD integration, i.e. how will one use it? Meanwhile, vendors will all say they are planning to be "compatible" to whatever the latest standard hits the trade journals. Ask when the offering will be available. Ask how the promised features will deliver the requested functionality, look for the shades of grey through the smoke and mirrors. "You get what you ask for", meaning, be an informed consumer.

Business Re-engineering

Until now the focus of this paper has been software re-engineering. Throughout our R^3 studies, we kept on encountering another "re" word that did not go away; it has become a prominent consideration. The objectives of sound business practice are to:

- Achieve a strategic and competitive position in the industry;

- Increase productivity;
- Respond rapidly and flexibly to deploy new or changing mission-critical functions;
- Be cost-effective;
- Serve the customer in the spirit of Total Quality Management.

Business re-engineering refers to modeling the organization business practices, and procedures to reach the above objectives.

How does technology support business needs?
Old ways of doing business are wedded to obsolete technology and that inefficiency puts an organization at a competitive disadvantage. Some critics of software re-engineering are concerned it perpetuates obsolete ways by means of improved technologies. Citing Paul A. Strassman, Department of Defense, the organization should view a modernization effort as a functional improvement of business processes, not a technology program.

For example, when looking for areas to improve, executives should look for unnecessary redundancy. Decentralization of operations or Total Quality Management principles also impact the service to the customer. Changes in the supporting technology which affect improvements to business process include consolidating and integrating data, providing seamless information flows and just-in-time data, and sharing reusable software components. However, advances in hardware have outpaced those of software architectures, leaving customers with increasingly greater expectations of what software should offer them. Strassman states the common glue for making a software engineering infrastructure feasible comes from a coherent approach to information management. Part of this approach "is contained in the principles of the separation of the roles of information providers from those of information customers."

Who are the customers at IRS? Broadly, IRS customers are the U.S.A. taxpayers, Congressional lawmakers and Department of Treasury. For developers of IRS software, our intermediaries are functional end-users in the IRS whose responsiblity is to implement and enforce tax law. For those creating the Software Development Environment and information technology infrastructure, customers are the developers of software.

How does a Software Re-engineering Framework support Business Re-engineering?
We believe an effective business process is the foremost priority and is absolutely essential for the good standing of the organization. Having established those major business prerequisites, we believe software re-engineering does have a role when an orderly transition into the new technology is necessary, as opposed to letting the new system replace the old wholesale.

Wholesale replacements are high risk, time-consuming and costly. Even though a new generation may not care for its legacy, the answer is not throwing away the existing system (like throwing out the baby with the bath water). Re-engineering should support a transition effort which determines the mapping, phased co-existence and synchronization between legacy and new systems, in two most critical areas: requirements or business functions, and production data migration. Creating data and/or process models for a legacy system, will promote understanding of what is being replaced by the new development life cycle. The effort will proceed with less risk than doing the new analysis blind or from scratch. The existing production operations are mission-critical and form the definitive baseline from which any changes must be managed.[12]

Not all re-engineering means migrating to new systems. CASE tools may be used to relieve the maintenance burden for existing systems. Support for maintenance during a long-term modernization effort should not be disregarded because the system will eventually be retired. Perhaps there is still cost benefit for CASE during its remaining useful life. CASE support for maintenance can be an enabling technology. While aiding analysis and maintenance it may also prepare a legacy system for migration to open systems, I-CASE, etc. It enables software process improvement and prepares the organization for the cultural readiness to use CASE.

Risk Management

There are business risks involved in developing software within estimated time and budget. There are significant benefits to evaluating current systems to determine what can be retired last (or not at all). Planning to utilize the current system and its strengths in a transition strategy or business plan reduces risk. Consider an organizational framework which approaches software development from a project management or risk-driven perspective, reference Slide 29.

The risk management techniques in the table, written by Barry Boehm, are part of a "spiral method for software development" which incorporates prototyping, evolutionary user and technical requirements, short requirement inspection and verification loops, and risk resolution. Spiral methods are analogous to down-sizing the software lifecycle away from top-down waterfall methods. It is risk-driven, where by comparison, information engineering is document-driven.

Risk management techniques fit in well with a range of software project situations. It fits well with the scenario-driven approaches of the redevelopment migration framework. The techniques are appropriate for high risk areas, when testing migration of re-engineered modules, and for internal software development (vs. contractually specified). Incremental prototyping and letting the end-user have a tool-assisted view of their specifications would be a suggested extension for some software development in TSM.

Recommendations for the Organization

This paper has been filled with commentary to those who are about to venture into the redevelopment world. Here we shall advise certain steps which we are now beginning to take; which, if we had to do it over, we would make our first priority. The recommendation which is the most important and prerequisite to all others is to establish an infrastructure to manage large-scale software development. Failure to

[12] Requests for changes and enhancements to maintenance systems need to be coordinated with new development work.

establish this infrastructure can weaken the effectiveness of the other recommendations which depend upon a stable foundation for software engineering.

Define and implement an infrastructure

Myth: "A software methodology and its supporting tool is sufficient for implementation."

Reality: A single I-CASE tool might satisfy needs for a standalone project. However, for large-scale software development and coordination (and IRS tips the scale!), the success of any technology hinges upon the infrastructure for software engineering. Management of the engineering process or a framework for software development projects, like an IPSE, Integrated Project Support Environment, should be understood and in place as standard operating procedures. The infrastructure supports management, IPSE/tool builders, and software developers; it supports cross-system development coordination and system redevelopment through to implementation (versus stopping short of supporting the full life cycle).

The greatest lesson we learned from our methods, tools, and prototype studies was that an organizational framework - infrastructure - is as critical, if not more, to the success of large scale software engineering than the technical nuts and bolts of CASE. For all three R^3 projects, the vendors presented findings citing the lack of IRS organizational readiness - explicitly citing lack of infrastructure - as an obstacle to continue applying R^3 approaches. However, all three vendors felt that R^3 approaches were critical to the success of TSM. Here the word infrastructure is used broadly; R^3 would be integrated with the entire organization, managing transition issues for large scale information technology. Continuing to perform prototypes would put the cart before the horse. Quality assurance procedures, change and configuration management, project management and short/long term procurement support should be in place before the development process begins. Those accountable for TSM should make infrastructure an immediate priority. Some R^3-related elements of an infrastructure for TSM are not in place yet, but are critical success factors, and are listed as follows:

- a transition strategy;
- objective- and requirement-driven software engineering processes;
- configuration management synchronizing old & new systems;
- traceability of requirements;
- reusable software components;
- procurement of tools for a software engineering environment or an IPSE.

Prepare an inventory of current systems. On an enterprise-wide scale, document the component names of system operations; identify the salient business purpose(s) for each main program. Create an integrated inventory of operations, showing relationships among system "schematic"/data flow components, programs, data files, users, developers, methods, etc. Later, for selected systems, further decompose to relate the data elements, processes and requirements. There are automated tools to support this effort. This inventory will be essential in evaluating enterprise business needs and providing a basis from which the portfolio analysis is made.

Conduct a business needs assessment/measurement phase. Identify software problems. Determine the extent to which the system or its separate components are functionally stable and represent future business and technical needs. These assessments are made at a system or program-level; source-level metrics are supported by automated tools. Metrics and measurements of various kinds[13] can be gathered to determine specific system problems. Tools can assist in identifying performance issues and maintenance-intensive code.

Develop criteria to select R^3 candidates. Using the business needs assessment information, graph current systems to portfolio analysis axes, judging technical and functional quality, resulting in a view of a high-level snapshot and prognosis for a class of problem types; reference Slide 10 and section above on Portfolio Analysis. Consider the feasibility of whether reusing or making improvements to existing software components supports business and technical objectives. Criteria for quality will, again, reflect future technical and organizational goals; e.g. whether a system's existing technical architecture matches closely to the future target architecture. Goals need not be grandiose, nor revolutionary; if incremental change is planned, the needs assessment and portfolio analysis would assist in planning the appropriate sequence of incremental improvements (see Transition Strategy below). For example, portfolio analysis may only model moving to a database from flat files, or to modular subroutines from large monolithic programs, both still on the same hardware platform.

Write a Transition Strategy. A transition strategy outlines low-risk target strategies over multiple phases in order to reach long term goals. Without a phased approach, those same strategies could have been high-risk.

There are two sets of transition strategies. The obvious one which first comes to mind is the transition schedule and strategy for end-user production systems from the base of existing legacy systems. The second transition strategy determines the schedule and changes to software engineering processes and environments while supporting maintenance and/or the transition to new end-user targets.

Prioritize suitable R^3 objectives for the entire enterprise's software engineering effort. Prioritizing expectations for re-engineering is iterative; the organization conducts its needs and current systems assessments, and balances business needs with short-term goals based on level-of risk, cost or time of effort. Objectives should be traced to customers, who own the requirements, and end-users. Objectives need to be identifiable by deliverables (measurable benefit or measurable barometer of objectives met).

Use R^3 opportunities immediately to support incremental organizational readiness: Model the business data of major IRS systems; make CASE-assisted, incremental, source-level improvements to assembler language programs; use current system data models not only to verify and validate data and processes of new development, but to serve as the baseline model for new development; and, allow maintenance programmers in areas using C language, or those supporting systems which will not be retired in the near term, the benefit of

[13] It is not within the scope of this paper to present a comprehensive taxonomy on the many forms of inspections, measurements, and metrics possible. However it is important, as with all processes, that measurements be taken which directly facilitate the user of the metric to benefit from the information, and help the user determine an appropriate software engineerng strategy.

using R^3 CASE to make incremental improvements. Determine a hierarchy of those activities, while coordinating dependencies.[14] Without having a transition strategy in place, the organization is at risk of completing a well-executed, technically-perfect project that neither supports organizational goals nor moves any closer to final objectives- and may be thrown away.

Choose a candidate project(s) targeting implementation. A re-engineering support program must provide some level of support to appropriate, selected candidates. Start out with a small and well-defined (manageable) scope in a proof-of-concept. Although the long-term modernization should be flexible as needed, a project's given objectives must be fixed, not moving targets. Beginning at the project-level, make a more detailed assessment and re-engineering plan. The project adapts software engineering processes to fit its situation, not one size-fits-all; reference Slides 9 & 11. Then, implement the project! Without implementation as a hard and fast goal, objectives may slide and findings may be tangential to use in real production. This is where the rubber meets the road for re-engineering concepts. It provides opportunities for technology transfer of software re-engineering processes to increasing numbers of technicians and managers. Successful use of re-engineering and technology transition are still on-the-outside-looking-in at IRS. There are many suitable projects at IRS which would demonstrate the pragmatism of re-engineering approaches. In fact, we argue, any development approach must by necessity include resolving transition issues.

Establish an R^3 team to support developers assisting with measurements, project planning, tools use and coordination with corporate modernization efforts. A large organization should be able to assemble a cadre of skilled software engineers and trouble shooters. A team supporting R^3 projects should report to a chief architect who is responsible for both an integrated, enterprise information systems architecture and the technical feasibility of the modernization solution. A permanent team demonstrates executive mandate, management support and authority to make technical decisions. The IRS has been performing its re-engineering work using a loosely matrixed organization, assigning staff part-time to the ad hoc effort. By creating a permanent team, resources and relationships can be consolidated to make the support for R^3 more stable and apply the technical expertise of personnel. It also capitalizes on the efficiency and effectiveness of a team of technical specialists, with experience in using tools and methods. Such a team can provide support to projects, facilitating technology transfer to the organization's grass roots increasingly over time. As technicians from the functional areas participate in projects with the R^3 team, they serve an evangelical purpose, carrying the experience and knowledge back to their maintenance environment. Organizations without in-house resources can hire consultants; however, the organization must devote some staff to work side-by-side with the consultant, otherwise the organization will have flat progress in technology transition.

[14] One example from the Department of Defense is in the CIM, Corporate Information Management, effort. DoD has a "reverse tree" in its strategy, where over time and several phases, DoD plans to merge and consolidate from ninety to eventually seven systems (Strassman).

Procure tools which meet the business needs, objectives and timeframes of the organization. Conduct a market survey. Identify the tools available to meet the business needs of the organization today. Again, know objectives first. Do not simply look for problems and fixes which fit a favorite tool or technology. Match the tools to needs. Look at the future technology trends in the industry, and develop flexible tool requirements and evaluation criteria with a long useful life. Markey surveys and evaluation criteria need to be maintained because CASE products and capabilities change at an extremely fast pace.[15] Apply the criteria and identify tools which are most appropriate as they are needed. Only purchase tools when they will be immediately used in a project.

Rationalize enterprise-wide data and standardize data names. This means eliminating redundant names for the same logical data structures and adopting a single standard name for analysis level elements across enterprise systems. Data name standardization across the enterprise is an central linchpin for those organizations which aim to integrate systems. Integrating systems may have several contexts: re-engineering to consolidate multiple systems to a single system and database; using data name standardization to better understand data interfaces among multiple systems; facilitating impact analysis and configuration management across the enterprise in response to change requests; assisting impact analysis or current systems analysis for moving like-data and related logic to new methodologies; creating a data dictionary; and using standard data names for new systems development.[16] These activities should be performed whether assisted by tools or not. Some tool support is available, although rationalizing and standardizing data names cannot be fully automated because understanding the meaning of the data usually requires the intelligent judgements of a domain expert.[17]

Technology Transition

Technology transition includes a "paradigm shift".[18] We do not think it means just learning to use CASE tools. We need to apply CASE technology, methodologies and perform business re-engineering for Information Technology, to find a better business process for software engineering. Part of the role of tool-assisted re-engineering is to

[15] David Sharon says market survey and evaluation criteria are 25% obsolete in three months and 50% obsolete in six months.

[16] The timing of having these data administration capabilities and procedures in place relative to the life cycle and implementation schedules of the customers (end users, software developers) makes a large difference in the purpose for which customers shall use standard data names and its impact on degrees of integration.

[17] Sometimes trying to force a process for data rationalization can be quite kluge-like using some tool which was not fully intended for that purpose. Our answer for tool-assisted data rationalization is a request to the vendor community: "Give us a good text editor!"

[18] A paradigm shift means that the way you see the world today forever renders obsolete the way you thought it was yesterday.

prepare the organization for "technology push", rather than being pulled along later, thereby accepting the fact that the tools today are in a fluid state or not mature (Mosley).

Our goal is to buy and use an Integrated Project Support Environment (IPSE). Although one stop shopping for an IPSE does not exist today, we shall assume a proactive role as a customer of IPSE, and integrated SEE (Software Engineering Environment). CASE standards for these environments are being developed by vendors and user groups. The opportunity to direct these efforts makes this time exciting.

Our participation in influencing the CASE market makes this time exciting, but it also can be problematic. Finding and purchasing the "best-in-class" tools, can be an elusive goal. Just when a decision is made, another product is released, including all the features requested. Dave Sharon calls it settling for "good enough" tools. The 80/20 rule. If you settle for 80% of exactly what you want, you may be able to move to solving your problems more quickly, as opposed to spending 80% of your time trying to find that last 20% functionality which may not exist in the market for years yet.

Marketing R3 Internally

Re-engineering provides a wide variety of opportunities to support business goals. But our experience in the IRS is that if those opportunities are not understood by management, if they cannot be demonstrated to specifically support critical organizational goals, this theory will be just another great idea that is not utilized. Many of the recommendations listed above identify the marketing strategy outlined here.

Be realistic. Remember, there is no magic, no automatic answers, no instant resolutions. Don't oversell it. Make sure both technicians and management understand that completely automated answers are not being proposed. Good techniques and reasonable benefits are the real advantages to re-engineering. Beware of presenting this as a "savior strategy".

Success is the best marketing strategy there is. As discussed in the recommendations above, identify a small project which targets production and is perceived as important. No amount of marketing will recover from a large failure. There will be a need to learn tools and methods. Providing a managable task in which staff can establish technical and procedural processes to follow will create that environment in which success is possible.

Continue to keep the benefits of this technology before the managers. Some will be convinced; others will remain skeptical. By providing reports which measure results and focus on the strategic benefits, they will serve as marketing vehicles. The comment from above, "don't oversell it" applies here as well.

Realities of R3

We have presented here some lessons learned from three years of struggling to understand this technology and its potential benefits. We discovered that of those lessons, the shattering of our myths and misconceptions, to include unrealistic expectations of the technology, were among the most important. With realistic understanding of the considerable possibilities using re-engineering techniques and methods, we could present a plan to include re-engineering in our transition strategy. The "realities" listed below represent those lessons learned, and the abandoning of those cherished myths.

Establish clear objectives - There are many solutions to attempt. Make the solutions fit the problems (not a favorite toolset).

No Silver Bullet - As stated several times in the paper, this technology is not an automated answer to all redevelopment problems. It involves hard work and, in many cases, manual work. But there are excellent tools and methods to support the analysis and evaluation work which must be done. Properly executed, a good return on investment can be demonstrated.

Embrace Business Re-engineering Early - Approaching the technology as a solution to business problems may lead to technical solutions simply providing a faster way of performing functions that no longer reflect the way the business operates. Not every organization is plagued with antiquated systems, fixing the business architecture in stone for many years. But a comprehensive software redevelopment will provide an opportunity to review the functional processes in light of new business objectives.

Get Experienced Guidance - Ordinarily, one immediately thinks of hiring contract personnel with experience in using tools and methods. And that is recommended. But don't forget in-house personnel, who have functional and organizational knowledge necessary to the success of any project. These organizational specialists should be made an equal participant with any contractors, building experience as they provide functional guidance. This partnership provides the means of gaining experience and building a team of resident experts.

Phased Change - Plan system implementation with organizational readiness. This "reality" reflects the integration of re-engineering phases with the IS strategy. In the case of the IRS, it is a transition strategy. Mapping specific re-engineering tasks to support strategic milestones serves as a roadmap for implementation. By identifying bite-sized tasks, the change can be planned in reasonable steps.

Balance R&D with ROI - Implement planned solutions; nothing gives a project motivation like putting it into production. There will necessarily be some studies which serve primarily to orient technical staff to tools and methods. However, proof-of-concept prototypes can be frustrating, if they only produce statistics and recommendations. The key is to identify production systems which can provide benefits to the organization - demonstrable return on investment (ROI).

Gather Case Study Results - Don't pursue objectives in the abstract. There is a tremendous amount of work that is being done in the industry, in both the private and federal sectors. Take advantage of the work that has already been done, lessons that have been learned. Jump-start corporate strategic planning with these valuable findings.

Transition is a major issue - When the objective of re-engineering is a redevelopment project, a critical step in the implementation of projects is a transition plan. Such a plan should include a mapping of old-to-new data, old-to-new processes, replacement strategy for old programs and systems, conversion of old files to new formats for historical reports, and replacement of hardware (some systems may be ported to the new hardware prior to conversion, in an attempt to limit the time two platforms have to be maintained). This transition period can be supported in significant ways with data re-engineering tools, slice-and-dice analysis tools to modularize the code for replacement and conversion and data conversion tools. Failure to plan this phase will result in a bottleneck during implementation, which would be unfortunate, as these are areas well-supported by automated tools.

Credits

- "Review of the Tax Systems Modernization of the Internal Revenue Service", National Academy Press, 1992.
- Barry W. Boehm. "A Spiral Model of Software Development and Enhancement". Richard Thayer, Tutorial: Software Engineering Project Management; The IEEE Computer Society Press, ©1988.
- Priscilla Fowler & Linda Levine. "Toward a Defined Process of Software Technology TRANSITION." American Programmer, Vol. 5 no. 3, March 1992.
- Chris Gane. "Extracting Business Rules"; lecture, Software Maintenance and Re-engineering Conference; DCI/CASE Trends; Washington, DC; 2/92.
- Al Kortesoja. "Redevelopment Engineering: A Management View." CASE Trends, 4/92 & 5/92; ©1992 Software Productivity Group.
- Daniel J. Mosley. "A Framework for Technology Innovation." American Programmer, Vol. 5 no. 3, March 1992.
- John Palmer. "The Big Crunch: Achieving Software Development Process Compression." Object Magazine; May/June, 1992.
- Lamont Phemister. "Requirements for Reverse Engineering: Squirrel & Butterflies for Functional Flow" CASE Trends, 10/92.
- David Sharon, CASE Associates, Inc. "Deliverable 27, White Paper on Tool Integration Standards," prepared for IRS under subcontract, CBIS Task 88; October, 1992.
- Paul A. Strassman. "Information Management Topics: The Policies, Processes, and Technologies of CIM." CrossTalk, The Journal of Defense Software Engineering; Number 37, October, 1992.
- William Ulrich. "Re-engineering: Defining an Integrated Migration Framework". CASE Trends, 11/90 - 6/91; ©1991 Software Productivity Group, Inc.

MYTHS AND REALITIES

Defining Re-engineering for a Large Organization

Sandra Yin
Office of Transition Management
Internal Revenue Service, ISM:TM:S
8405 Colesville Rd., Suite 300
Silver Spring, MD 20910-3312
301/427-0151; fax 301/427-0276

Introduction

- Internal Revenue Service's Tax Systems Modernization
- Reverse engineering, re-engineering, reuse = "R^3"
- Three R^3 projects - findings and recommendations
  - IRS corporate assessments, R^3 methods
  - Proof-of concept prototype
  - Market Survey of off-the-shelf tools
- Role of Infrastructure, Business Re-engineering, and Technology Transition

Concepts, Content and Context: IRS Assessment and R^3 Methods

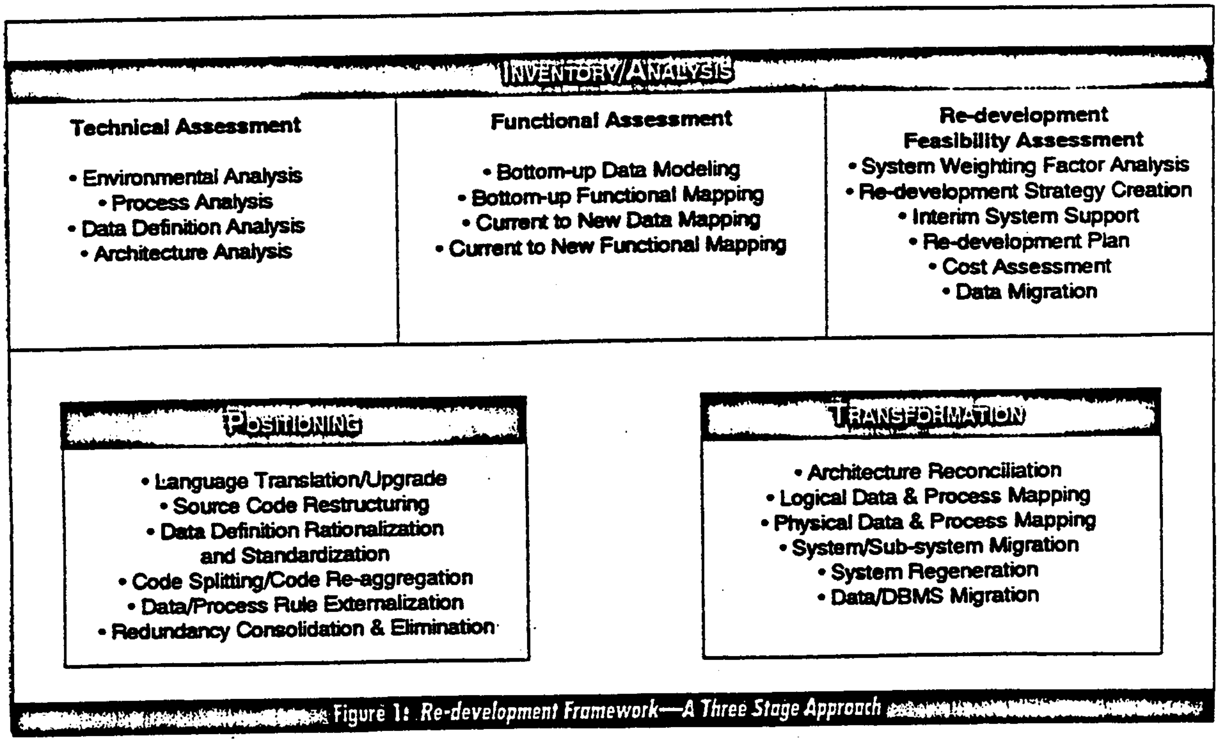
Objectives

- Taxonomy of R^3 terms and definitions
- Reverse engineering and re-engineering methods
- Corporate assessment on broad needs
- Project plans for four executive-selected, IRS systems

Software Re-engineering Taxonomy

- Re-engineering
  - IEEE: combined processes encompassing reverse and forward engineering, resulting in a "new" system
  - Guide: improving current systems without impacting current functions, technical platforms or archictectures
  - Ulrich: A combination of tools and techniques that facilitate the analysis, improvement, redesign and reuse of existing software systems to support changing information requirements
- Reverse engineering
  - process taking existing system and migrating it back to a higher level of abstraction
- Reuse
  - applying knowledge about one system to another system; sharing software components, requirements, and effort of maintenance

A Framework for Redevelopment



INVENTORY/ANALYSIS

Technical Assessment

- Environmental Analysis
- Process Analysis
- Data Definition Analysis
- Architecture Analysis

Functional Assessment

- Bottom-up Data Modeling
- Bottom-up Functional Mapping
- Current to New Data Mapping
- Current to New Functional Mapping

Re-development Feasibility Assessment

- System Weighting Factor Analysis
- Re-development Strategy Creation
- Interim System Support
- Re-development Plan
- Cost Assessment
- Data Migration

POSITIONING

- Language Translation/Upgrade
- Source Code Restructuring
- Data Definition Rationalization and Standardization
- Code Splitting/Code Re-aggregation
- Data/Process Rule Externalization
- Redundancy Consolidation & Elimination

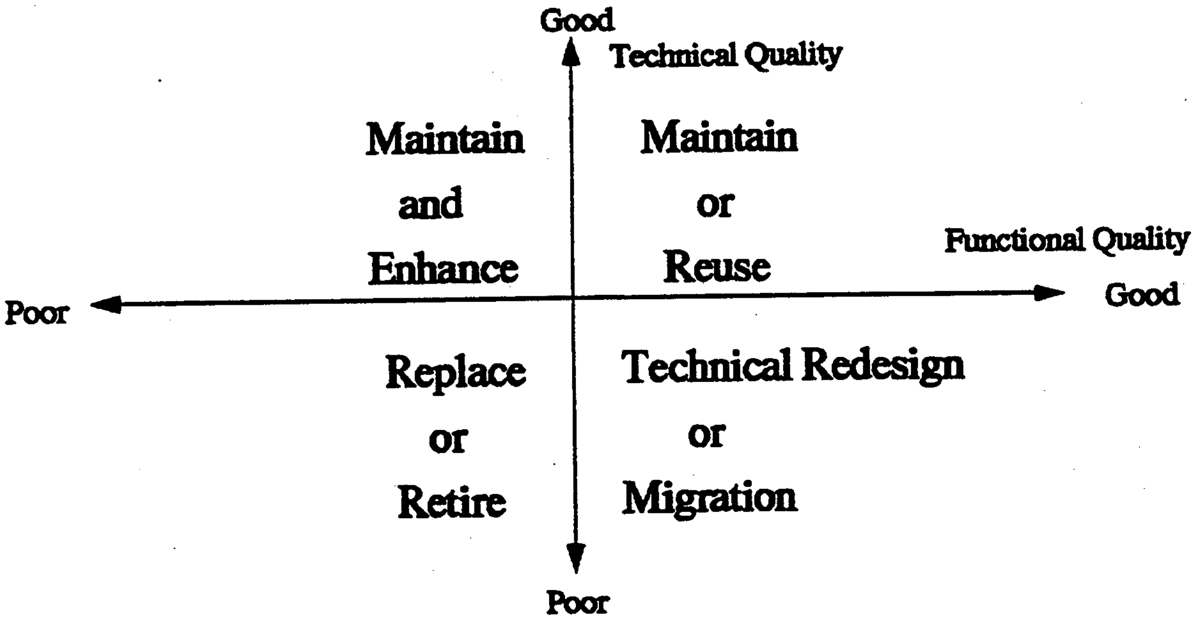
TRANSFORMATION

- Architecture Reconciliation
- Logical Data & Process Mapping
- Physical Data & Process Mapping
- System/Sub-system Migration
- System Regeneration
- Data/DBMS Migration

Figure 1: *Re-development Framework—A Three Stage Approach*

Source: William Ulrich, "Re-engineering: Defining an Integrated Migration Framework".
11/90 - 6/91 CASE Trends, ©1991 Software Productivity Group, Inc.

Portfolio Analysis



- An Enterprise-level Current Systems Assessment
- Not done at IRS, must be done
- Quality: Stable and represents requirements
- Categorize sets: Mission-critical, duplicate application functions, hardware platforms

Up, Over, and Down

(or, Look Before You Leap) - Applied Use of Methods

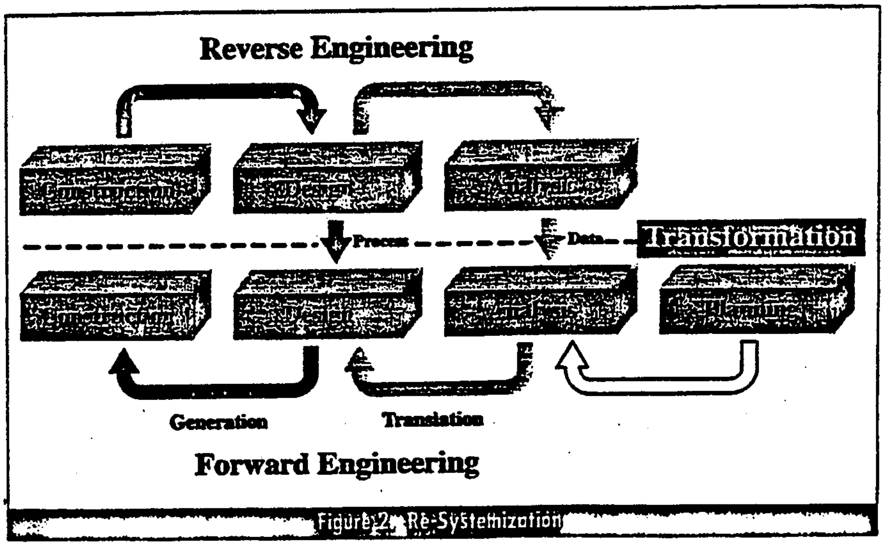


Figure 2. Re-Systemization

- Objective: Populating an I-CASE tool with existing system components
- Source: Al Kortesoja, "Redevelopment Engineering: A Management View". 4/92 & 5/92 CASE Trends, ©1992 Software Productivity Group.

Business Re-engineering

- Achieve strategic and competitive position
- Increase productivity
- Flexibility to respond rapidly and deploy new or changing mission-critical functions
- Be cost-effective
- Serve the customer - Total Quality Management

How does technology support business needs?

Citing Paul A. Strassman, DoD/DIS

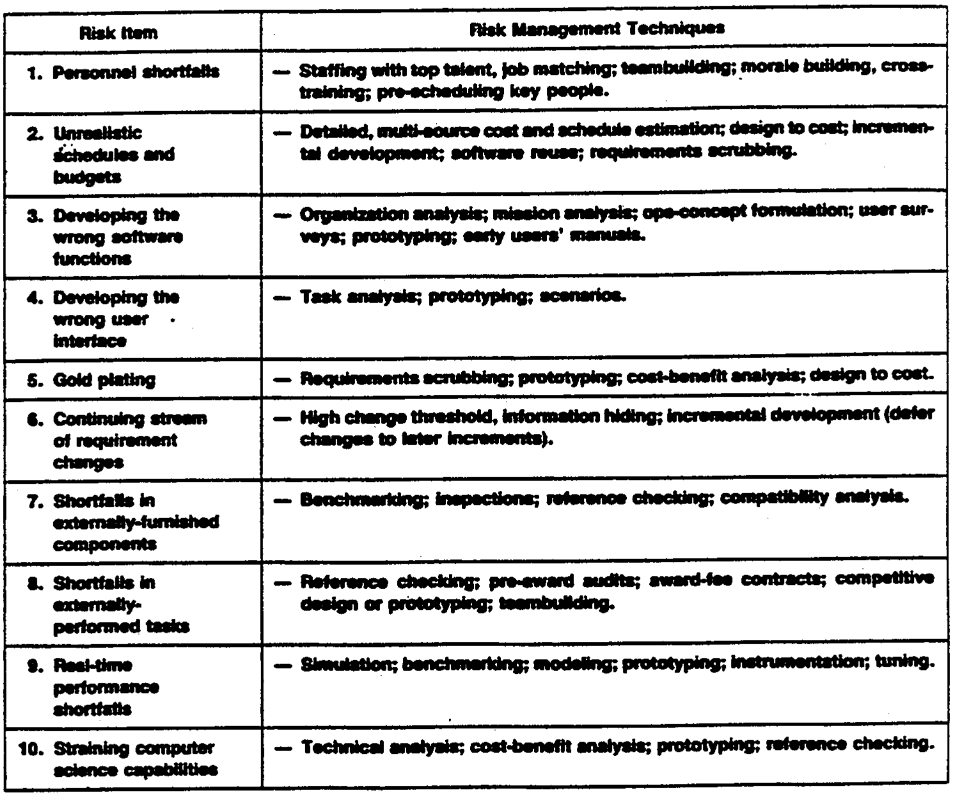
- Do not perpetuate obsolete ways by means of improved technologies
- Functional improvement of business processes vs. technology program
- Reduce unnecessary reduncy
- Seamless information flows, just-in-time data
- Decentralization
- Total Quality Management
- Common glue for making the infrastructure feasible comes from a coherent approach to information management.

How does Software Re-engineering Framework support Business Re-engineering?

- Wholesale replacements time-consuming and costly
- Determine co-existence and synchronization between legacy and new systems
  - Requirements or business functions
  - Data migration
- Use of tools as enabling technology
  - Aid analysis and maintenance
  - Prepare for migration to open systems
- Existing production is mission-critical, The Bottom Line baseline

Risk Management

Figure 3: A Prioritized Top Ten List of Software Risk Items



Risk Item	Risk Management Techniques
1. Personnel shortfalls	— Staffing with top talent, job matching; teambuilding; morale building, cross-training; pre-scheduling key people.
2. Unrealistic schedules and budgets	— Detailed, multi-source cost and schedule estimation; design to cost; incremental development; software reuse; requirements scrubbing.
3. Developing the wrong software functions	— Organization analysis; mission analysis; ops-concept formulation; user surveys; prototyping; early users' manuals.
4. Developing the wrong user interface	— Task analysis; prototyping; scenarios.
5. Gold plating	— Requirements scrubbing; prototyping; cost-benefit analysis; design to cost.
6. Continuing stream of requirement changes	— High change threshold, information hiding; incremental development (defer changes to later increments).
7. Shortfalls in externally-furnished components	— Benchmarking; inspections; reference checking; compatibility analysis.
8. Shortfalls in externally-performed tasks	— Reference checking; pre-award audits; award-fee contracts; competitive design or prototyping; teambuilding.
9. Real-time performance shortfalls	— Simulation; benchmarking; modeling; prototyping; instrumentation; tuning.
10. Straining computer science capabilities	— Technical analysis; cost-benefit analysis; prototyping; reference checking.

Source: Barry W. Boehm, "A Spiral Model of Software Development and Enhancement". ©1988

Tips, Tricks and Traps: The R^3 Prototype

Objectives

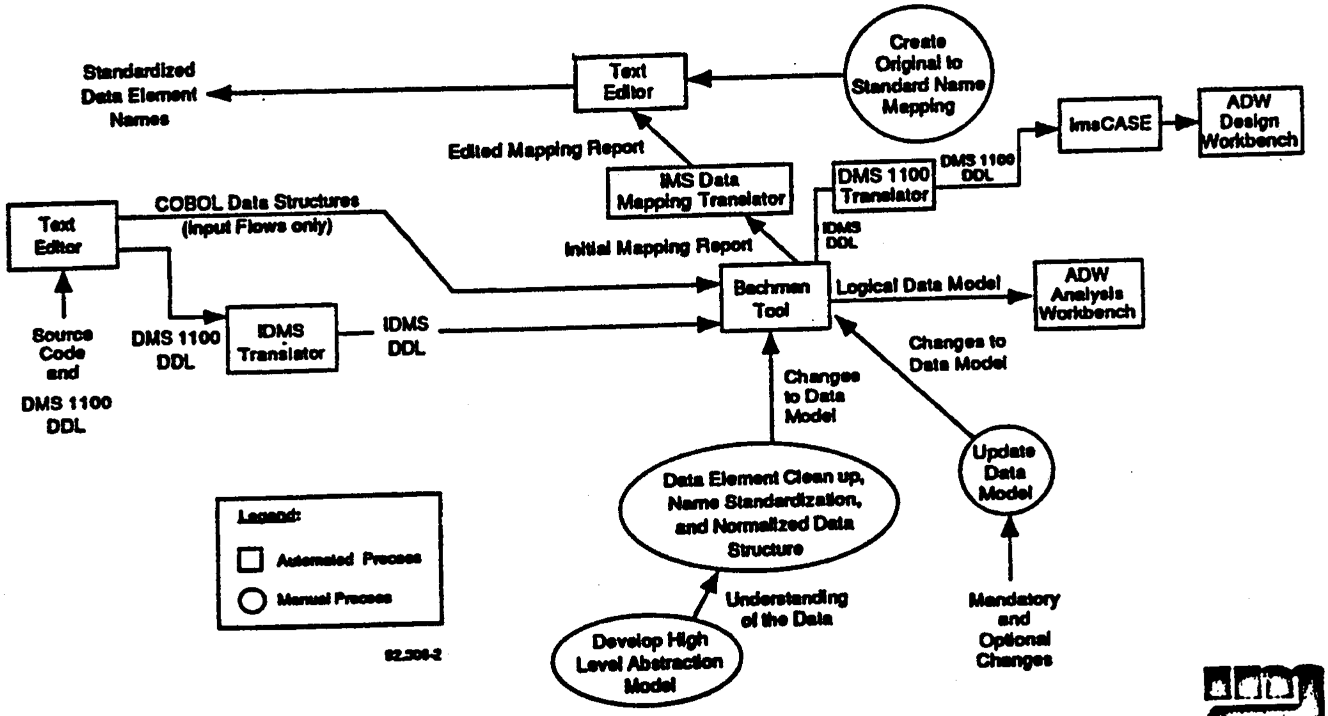
- Use integrated CASE toolset from source to target
- Single, integrated repository
- Data standardization
- Verify migration to CASE first, then try enhancements
- Technology transfer

Tips, Tricks and Traps: The R^3 Prototype

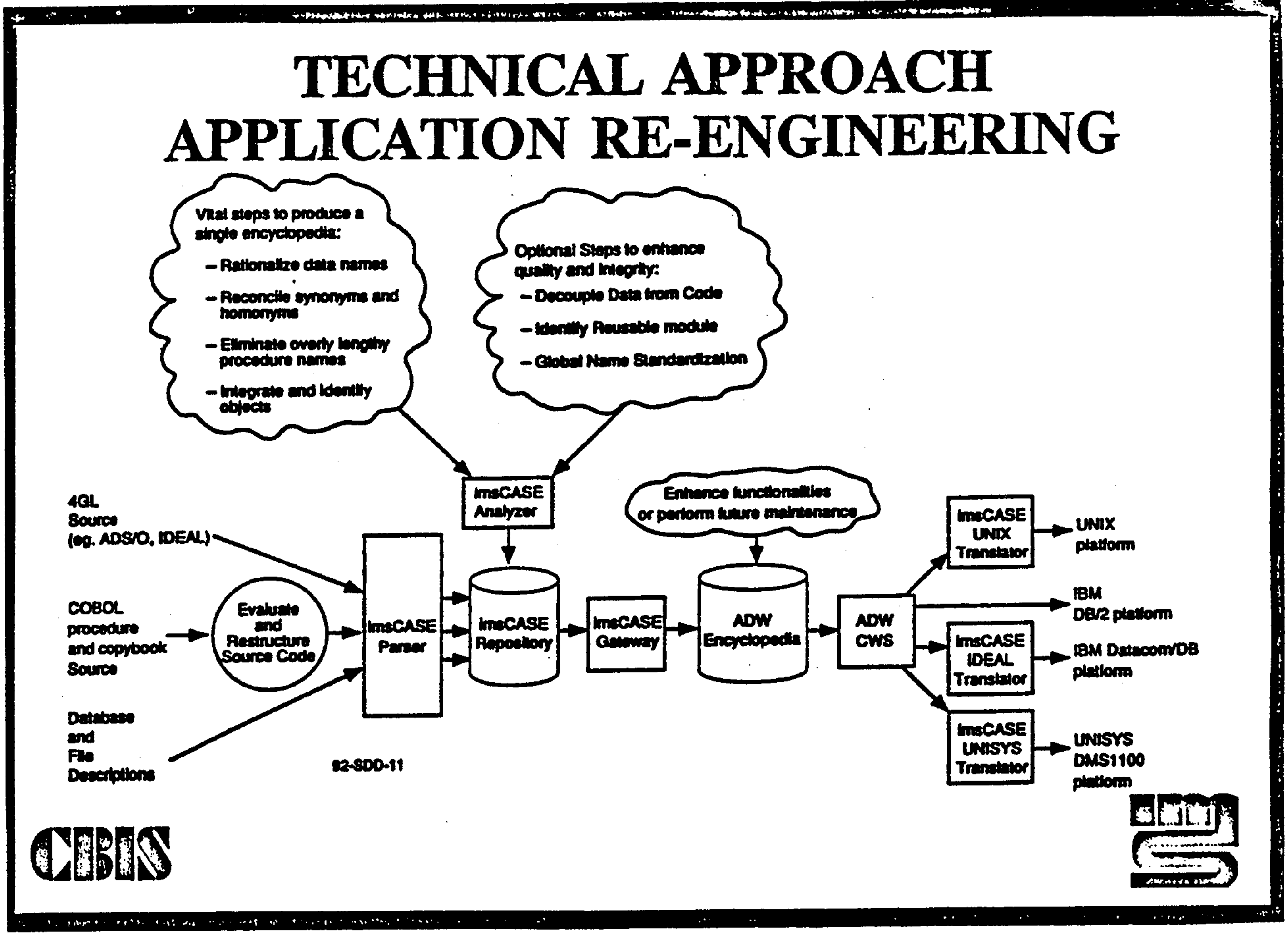
Findings

- Successful code generation, tracing to design level in CASE
- Populate CASE tool with reuseable objects
- Inventory/analysis assessments should drive objectives
- Clean-up existing source before populating CASE
- Services with tools needed for niche specialities

TECHNICAL APPROACH DATA RE-ENGINEERING


Standardized Data Element Names
Text Editor
Create Original to Standard Name Mapping
ImsCASE
ADW Design Workbench
Edited Mapping Report
IMS Data Mapping Translator
DMS 1100 Translator
DMS 1100 DDL
Text Editor
COBOL Data Structures (Input Flows only)
IDMS DDL
Initial Mapping Report
Bachman Tool
Logical Data Model
ADW Analysis Workbench
Source Code and DMS 1100 DDL
DMS 1100 DDL
IDMS Translator
IDMS DDL
Changes to Data Model
Changes to Data Model
Update Data Model
Data Element Clean up, Name Standardization, and Normalized Data Structure
Understanding of the Data
Develop High Level Abstraction Model
Mandatory and Optional Changes
Legend:
□ Automated Process
○ Manual Process

CBIS



TECHNICAL APPROACH APPLICATION RE-ENGINEERING

Vital steps to produce a single encyclopedia:

- Rationalize data names
- Reconcile synonyms and homonyms
- Eliminate overly lengthy procedure names
- Integrate and identify objects

Optional Steps to enhance quality and integrity:

- Decouple Data from Code
- Identify Reusable module
- Global Name Standardization

Features and Futures: Automated Tool Market Survey

Objectives

- Re-engineering methods are baseline for tool classifications.
- Interview representative maintenance and IEM community
- Apply tool evaluation criteria against an existing database
- Determine best-in-class, and integrated toolset.
- Visit top 15 vendors to discuss tool futures.
- Write white paper on standards and tool integration.

Features and Futures: Automated Tool Market Survey

Interview results: IRS R^3 Objectives

1. Creation of an Inventory of All Current Systems

2. Faster Software Maintenance While Capturing Information About the Current Systems

3. Current System Verification and Validation of Business Area Analyses Created by the Information Engineering Group

4. Information Extracted From Current Systems to Assist in the Development of:

   a. New BAAs
   b. An R^3 I-CASE Repository for Forward Engineering

5. Transition to New Target Environments Supporting the Standards Based Architecture

Source: David Sharon, CASE Associates; CBIS Task #88

Features and Futures: Automated Tool Market Survey

Findings

- Tool Integration
- Multi-Vendor R3 Tool Set Examples
- Commercially Available Integration Environments
- Transition Challenge: Managing Multiple Tools

Multi-Vendor R3 Tool Set Examples

R^3 Tool Classes

Tool Set Examples	1.0 Existing Systems	2.0 Repository Load	3.0 New/Replace Systems	4.0 Repositories	5.0 Integrated Tool Sets	6.0 Testing/ Validation	7.0 Software/ Proj Mgt
Tool Set 1	Bachman and Arrae	Arrae	IEF	IEF	-custom-	None	None
Tool Set 2	Adpac and Compuware	Custom Bridge	ADW	ADW	-custom-	Compu-ware	None
Tool Set 3	Adpac	Infospan	ADW	Infospan, ADW	-custom-	None	None
Tool Set 4	Cadre to Adpac	Infospan	Bachman	Infospan, Bachman	-custom-	None	None
Tool Set 5	Cadre or Procase	Cadre or Procase	Cadre or IDE	Cadre or IDE	Softbench	Many	Softool
Tool Set 6	Ernst & Young Adpac Bachman	RE/Toolset Reltech	ADW	ADW	-custom-	Viasoft	None

Source: David Sharon, CASE Associates; CBIS Task #88

Tool Integration

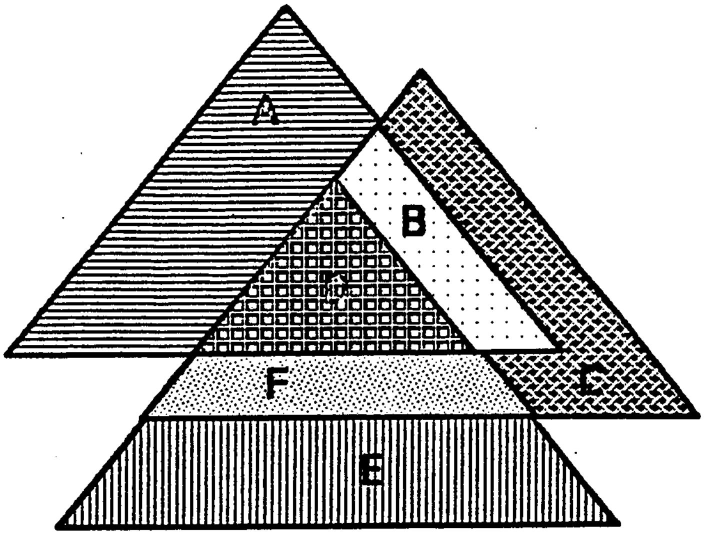
- **Standards-driven open systems**
- **Mandatory to integrate -**
  - **Process management**
  - **Presentation**
- **Optional/as-needed to integrate -**
  - **Data**
  - **Control services**

Commercially Available Integration Environments

Vendor	Product	Primary Emphasis	Environment	CASE Tools Supported
BrownStone Solutions	Data Dictionary/ Solution	Data	MS-DOS, OS/2, DB2/MVS	IEW/ADW, Bachman
InfoSpan	CaseSpan		MS-DOS, OS/2, UNIX	IEW/ADW, Excelerator, PM/SS
Reltech	DB Excel		MS-DOS, OS/2, DB2/MVS	IEW/ADW, Excelerator, Bachman
Software One	Exchange		MS-DOS	IEF, IEW, Excelerator, Auto-Mate Plus, Systems Engineer, Bachman/Analyst, Datamanager, ICL DDS, Oracle, Ingres, Telon
Atherton Technology	Software Backplane	Control	VAX/VMS, Ultrix, MVS, SunOS	Over 50 tools
Digital Equipment Corporation	Cohesion		VMS, Ultrix	Teamwork, Excelerator, Software through Pictures, Netron/CAP
Hewlett-Packard	SoftBench		UNIX	Over 60 tools
IBM	SDE/6000		UNIX	Over 50 tools
SFGL	EAST (European Advanced Software Technology)		UNIX	Any PCTE-compliant tool
Softlab	Maestro II		MS-DOS, UNIX	MicroFocus Cobol Workbench
Delphi Group	SEE System	Process	MS-DOS, OS/2 PM	Any
Rapid System Development	HyperAnalyst		OS/2, MVS/DB2	Bachman

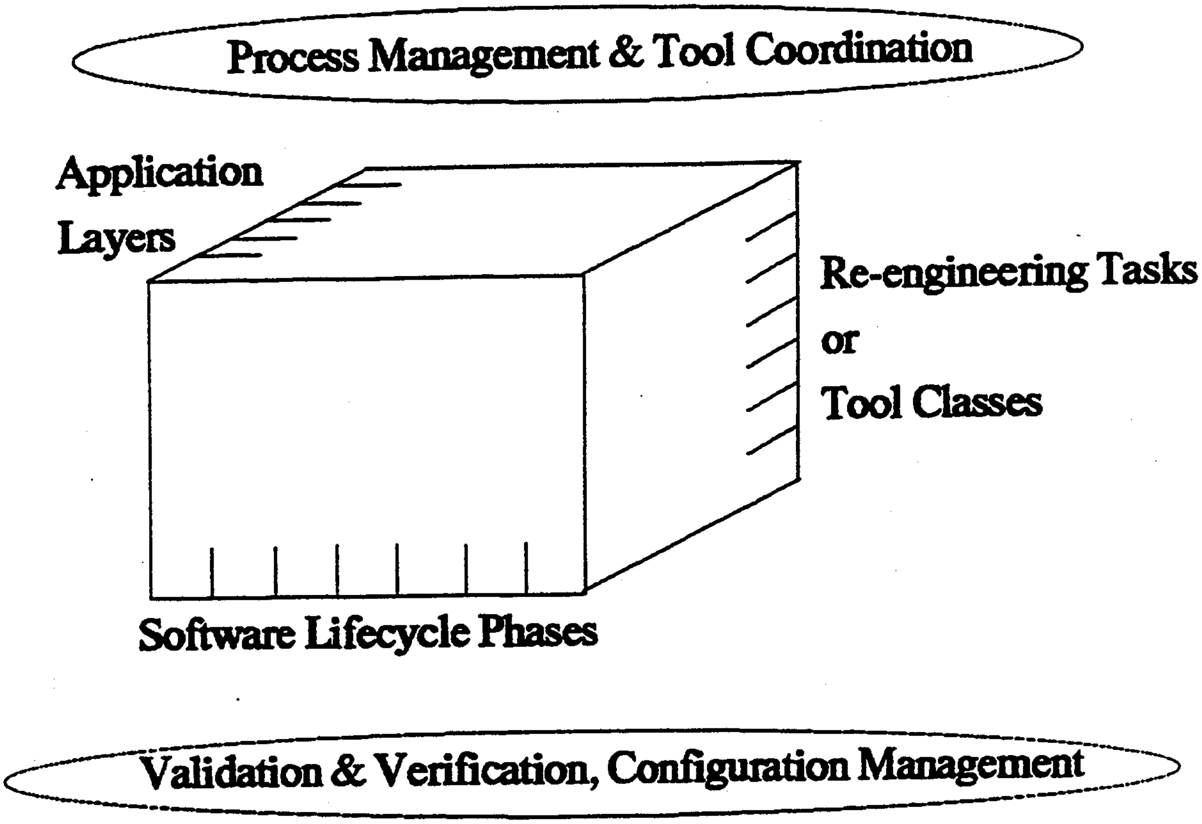
Source: David Sharon, CASE Associates; CBIS Task #88

Transition Challenge: Managing Multiple Tools



A = Current Systems, all existing production
B = Change requests, testing, field operations, database admin
C = The Integrator - Configuration Mgt, Enterprise & Mgt Info
D = Requirements, Rationalized Data/Code, CSA, data migration
E = ICASE development, forward engineering
F = ICASE verification & validation, model management

Roll up your shirt sleeves: R^3



Process Management & Tool Coordination

Application Layers

Re-engineering Tasks
or
Tool Classes

Software Lifecycle Phases

Validation & Verification, Configuration Management

Recommendations for the Organization

- Define and implement an infrastructure
- Establish a R^3 team
- Inventory current systems
- Conduct business needs asssessment
- Select R^3 projects from Portfolio Analysis
- Procure tools
- Rationalize, standardize corporate-wide data

Define and implement an infrastructure

Critical Success Factors for Tax Systems Modernization

- Objective- and requirement-driven processes
- Configuration Management synchronizing old & new
- Traceability of requirements
- Data Standardization
- Reuse software components
- Procurement

Technology Transition

- ”Paradigm Shift” means business-reengineering for Information Technology
- Prepare organization for ”technology push”
- Settle for ”good enough” vs. best-in-class tools
- Have role as customer of IPSE (Integrated Project Support Environment) and CASE

Credits

- Julia McCreary, Barbara Shammas, & many IRS colleagues.
- Barry W. Boehm. "A Spiral Model of Software Development and Enhancement". Richard Thayer, Tutorial: Software Engineering Project Management; The IEEE Computer Society Press, ©1988.
- Priscilla Fowler & Linda Levine. "Toward a Defined Process of Software Technology TRANSITION." American Programmer, Vol. 5 no. 3, March 1992.
- Al Kortesoja. "Redevelopment Engineering: A Management View." CASE Trends, 4/92 & 5/92; ©1992 Software Productivity Group.
- Daniel J. Mosley. "A Framework for Technology Innovation." American Programmer, Vol. 5 no. 3, March 1992.
- John Palmer. "The Big Crunch: Achieving Software Development Process Compression." Object Magazine; May/June, 1992.
- Lamont Phemister. "Requirements for Reverse Engineering: Squirrel & Butterflies for Functional Flow" CASE Trends, 10/92.
- David Sharon, CASE Associates, Inc. "Deliverable 27, White Paper on Tool Integration Standards," prepared for IRS under subcontract, CBIS Task 88; October, 1992.
- Paul A. Strassman. "Information Management Topics: The Policies, Processes, and Technologies of CIM." CrossTalk, The Journal of Defense Software Engineering; Number 37, October, 1992.
- William Ulrich. "Re-engineering: Defining an Integrated Migration Framework". CASE Trends, 11/90 - 6/91; ©1991 Software Productivity Group, Inc.